Method for the automatic address generation of modules within clusters comprised of a plurality of these modules

Patent Number: ☐ US6038650
Publication date: 2000-03-14
Inventor(s): MUENCH ROBERT (DE); VORBACH MARTIN (DE)
Applicant(s): PACT INF TECH GMBH (DE)
Requested Patent: ☐ DE19704044
Application Number: US19970946999 19971008
Priority Number(s): DE19971004044 19970204
IPC Classification: G06F9/38
EC Classification: G06F15/78R
Equivalents:

Abstract

A method of automatic address generation by units within clusters of a plurality of such units in which individual configurable elements of a unit can be addressed. It is thus possible to address the individual elements directly for reconfiguration. This is a prerequisite for being able to reconfigure parts of the unit by an external primary logic unit without having to change the entire configuration of the unit. In addition, the addresses for the individual elements of the units are automatically generated in the X and Y directions, so that the addressing scheme represents the actual arrangement of units and configurable elements. Furthermore, manual allocation of addresses is not necessary due to automatic address generation. In accordance with the present invention, a cluster is provided with a number of configurable units, each having two inputs for receiving the X address of the last element of the preceding unit in the X direction (row) and the Y address of the last element of the preceding unit in the Y direction (column) and having two outputs to relay to the next unit the position of the last element of the unit in the X direction and in the Y direction.

(19) BUNDESREPUBLIK DEUTSCHLAND

DEUTSCHES PATENTAMT

(12) **Offenlegungsschrift**

(10) **DE 197 04 044 A 1**

(51) Int. Cl.[6]:
G 06 F 9/44
G 06 F 15/80

(21) Aktenzeichen: 197 04 044.6
(22) Anmeldetag: 4. 2. 97
(43) Offenlegungstag: 13. 8. 98

(71) Anmelder:
Pact Informationstechnologie GmbH, 81545 München, DE

(72) Erfinder:
Vorbach, Martin, 76149 Karlsruhe, DE; Münch, Robert, 76149 Karlsruhe, DE

(56) Entgegenhaltungen:

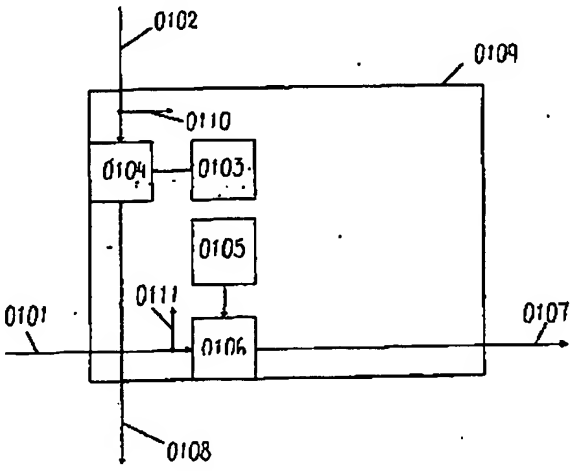
DE	44 16 881 A1
US	55 55 401
WO	96 04 603 A1

Die folgenden Angaben sind den vom Anmelder eingereichten Unterlagen entnommen

Prüfungsantrag gem. § 44 PatG ist gestellt

(54) Verfahren zur automatischen Adressgenerierung von Bausteinen innerhalb Clustern aus einer Vielzahl dieser Bausteine

(57) In Verbindung mit einem Verfahren zur dynamischen Adreßgenerierung konfigurierbarer Bausteine, mit einer zwei- oder mehrdimensionalen Zellanordnung (zum Beispiel FPGAs, DPGAs, DFPs o. ä.) und deren Zellen (konfigurierbaren Elemente) wird vorgeschlagen, daß
1. jeder Baustein einen Adreßeingang für jede Dimension besitzt, durch den er seine Basisadresse, die die Adresse der ersten Zelle darstellt, erhält, wobei dann
2. anhand der Basisadresse die Baustein-internen Zelladressen (Adressen der konfigurierbaren Elemente) berechnet werden, über die eine Ladelogik die konfigurierbaren Elemente anspricht, und wobei
3. über Adreßausgänge die Adresse der letzten Zelle plus eins, sowie je nach Ausgestaltung (siehe Unteransprüche) die Bausteinadresse plus eins, an die Nachfolgebausteine weitergegeben wird und deren Grundadresse darstellt.



DE 197 04 044 A 1

DE 197 04 044 A 1

Beschreibung

1. Hintergrund der Erfindung

1.1 Stand der Technik

Werden programmierbare Bausteine (FPGAs, DPGAs, DFPs (gemäß Offenlegung DE 44 16 881 A1)), im folgenden der Einfachheit halber unter dem Begriff "Bausteine" zusammengefaßt, zu einem Cluster zusammengefügt, gibt es zwei Arten, die Programmierung vorzunehmen. Ein Cluster ist eine mehrdimensionale vernetzte Anordnung von Bausteinen oder Bausteingruppen.

- Für jeden Baustein steht eine Ladelogik zur Verfügung, mit deren Hilfe der Baustein programmiert wird. Das heißt, eine Ladelogik adressiert einen Baustein des Clusters. Diese Ladelogik kann als EPROM oder als ein Rechnersystem ausgeführt sein. Die Daten können in serieller Form oder parallel zum Baustein übertragen werden. Bei der parallelen Datenübertragung werden die Daten im Baustein in einen seriellen Bitstrom umgewandelt, der den Baustein konfiguriert.
- Für alle Bausteine des Clusters steht nur eine Ladelogik zur Verfügung. Die einzelnen Bausteine sind in der Art einer Daisy-Chain zusammengeschaltet. Die Programmierung erfolgt durch einen seriellen Bitstrom. Dabei wird von der Ladelogik nur der erste Baustein adressiert und die Daten werden durch eine serielle Leitung zu den Bausteinen des Clusters geschickt. Die Daten werden als Bitstrom durch alle Bausteine des Clusters hindurchgeleitet und konfigurieren so deren programmierbare Elemente. Die Übertragung der Konfigurationsdaten kann wieder in serieller oder paralleler Form erfolgen. Die Ladelogik kann wiederum als EPROM oder als ein Rechnersystem ausgeführt sein.
- Bei der Verwendung eines Rechnersystems besteht die Möglichkeit die Adressleitungen für ein Chipselect zu verwenden, um einzelne Bausteine getrennt zu adressieren und zu konfigurieren.

Bei einem Parallelrechner wird jedem seiner Prozessoren eine feste Adresse zugewiesen, über die eine Adressierung erfolgt.

1.2 Probleme

Die bisherigen Verfahren zur Adressierung eines Bausteins weisen eine Reihe von Problemen und Schwächen auf.

- Durch die Art der Adressierung wird die tatsächliche Anordnung der Bausteine nicht repräsentiert.
- Es ist nicht möglich, ein einzelnes Element eines beliebigen Bausteins im Cluster zu adressieren, um eine Umkonfiguration vorzunehmen. Es kann immer nur ein gesamter Baustein adressiert und im Gesamten umkonfiguriert werden.
- Durch die feste Zuordnung der Adressen der einzelnen Prozessoren bei einem Parallelrechner, ergibt sich eine statische Zuordnung der Adressen. Außerdem entsteht ein hoher Aufwand bei der Vergabe der Adressen für die einzelnen Prozessoren.

1.3 Verbesserung durch die Erfindung

Mit Hilfe der Erfindung lassen sich einzelne konfigurierbare Elemente eines Bausteins adressieren. Damit ist es möglich, die einzelnen Elemente direkt für eine Umkonfigurierung anzusprechen. Dies ist eine Voraussetzung, um Teile des Bausteins durch eine externe Ladelogik umkonfigurieren zu können und nicht die gesamte Konfiguration des Bausteins verändern zu müssen. Außerdem werden die Adressen für die einzelnen Elemente der Bausteine automatisch in X- und Y-Richtung generiert, so daß die tatsächliche Anordnung der Bausteine und konfiguriebaren Elemente repräsentiert wird. Durch die automatische Adressgenerierung ist eine manuelle Vergaben der Adressen nicht notwendig.

Die Einzelheiten und besondere Ausgestaltungen, sowie Merkmale des erfindungsgemäßen Adressgenerierung sind Gegenstand der Patentansprüche.

2. Beschreibung der Erfindung

2.1 Übersicht über die Erfindung, Abstrakt

Ein Cluster enthält eine Vielzahl an konfigurierbaren Bausteinen, die jeweils zwei Eingänge zum Empfang der X-Adresse des letzten Elements des vorhergehenden Bausteins in X-Richtung (Reihe) und der Y-Adresse des letzten Elements des vorhergehenden Bausteins in Y-Richtung (Spalte) besitzen, sowie jeweils zwei Ausgänge, um die Position des letzten Elements des Bausteins in X-Richtung und um die Position des letzten Elements in Y-Richtung dem nachfolgenden Baustein weiterzugeben.

Der Ursprung des Koordinatensystems liegt für diese Schrift dabei in der linken oberen Ecke des Clusters, so daß die X-Adresse nach rechts und die Y-Adresse nach unten hin größer wird. Selbstverständlich kann für den Ursprung des Koordinatensystems auch eine andere Position gewählt werden.

Jeder der Bausteine des Clusters enthält eine Logik, zur Berechnung der X- und Y-Adresse der letzten Zelle des Bausteins. Diese Logik besteht aus jeweils einem Register für die X- und Y-Adresse, in dem die maximale Zahl der Elemente in X- und Y-Richtung des Bausteins gespeichert ist und einem Addierer, der die Adresse des letzten Elements des Bausteins in X- und Y-Richtung berechnet und über die beiden Ausgänge weitergibt. Jeder Baustein enthält wiederum eine

Vielzahl an konfigurierbaren Elementen, die eine Adresse zugewiesen bekommen und mit dieser Adresse innerhalb des Bausteins und des Clusters adressierbar sind. Die Adresse des Elements wird aus der Adresse des vorhergehenden Elements in X-Richtung und in Y-Richtung berechnet. Dadurch entsteht ein linearer Adressraum, in dem alle Elemente der Bausteine des Clusters adressierbar sind. Die Adresse des vorhergehenden Elements kann von einem Element dieses Bausteins oder des vorhergehenden Bausteins stammen. Um nun innerhalb eines Clusters ein Element in einem Baustein adressieren zu können, enthält jedes Element einen Vergleicher, in dem überprüft wird, ob eine von außen kommende Adresse mit der Adresse des Elements übereinstimmt.

2.2 Detailbeschreibung der Erfindung

Eine Vielzahl von konfigurierbaren Bausteinen wird zu einem Cluster zusammengefügt. Die konfigurierbaren Bausteine sind in Spalten und Reihen angeordnet. In den einzelnen Bausteinen ist wiederum eine Vielzahl von konfigurierbaren Elementen enthalten.

Jeder Baustein des Clusters steht über zwei Busse mit seinen Nachbarn in den Reihen und über zwei Busse mit seinen Nachbarn in den Spalten in Verbindung. Dabei dient ein Bus zum Empfang der Adressdaten vom vorhergehenden Nachbarn und ein Bus zum Versenden der Daten zum nachfolgenden Nachbarn. Über diese Leitungen werden die X-Adresse (Position innerhalb der Reihe) des letzten Elements des vorhergehenden Bausteins in X-Richtung und die Y-Adresse (Position innerhalb der Spalte) des letzten Elements des vorhergehenden Bausteins in Y-Richtung übertragen. Dies kann über eine serielle Leitung oder über mehrere Leitungen, deren Anzahl abhängig von der Anzahl der Bausteine des Clusters und den in den Bausteinen enthaltenen Elementen ist, parallel geschehen.

Eine Logik innerhalb des Bausteins berechnet die Adresse des letzten Elements des Bausteins in X-Richtung und des letzten Elements des Bausteins in Y-Richtung. Dazu wird zu der in einem Register gespeicherten Anzahl der Elemente in X-Richtung (Y-Richtung) die Adresse des letzten Elements in X-Richtung (Y-Richtung) des vorhergehenden Bausteins durch einen Addierer hinzugezählt. Damit ergibt sich die Adresse, die der maximalen Ausdehnung der Elemente in X,Y-Richtung innerhalb des Bausteins entspricht. Diese Adresse wird an den nächsten Baustein weitergegeben. Es werden jeweils ein Register und ein Addierer pro Richtung benötigt. Die Bausteine der ersten Spalte und der ersten Reihe des Clusters, müssen durch ein externes Signal die Basisadresse für die Elemente zugewiesen bekommen, da sie dort keine direkten Vorgänger besitzen.

Für die in den Bausteinen enthaltenen konfigurierbaren Elemente wird eine Adresse aus der Adresse des vorhergehenden Elements berechnet. Dieses vorhergehende Element kann im selben oder in dem vorhergehenden Baustein liegen. Zur Übertragung der Adressen sind auch die Elemente analog zu den Bausteinen, durch eine oder mehrere Leitungen miteinander verbunden. Auch hier kann die Übertragung der Adresse, wie bei den Bausteinen, seriell oder parallel erfolgen. Bei der Berechnung der Elementadresse wird zur Adresse des vorhergehenden Elements in X-Richtung (Y-Richtung) eine Eins hinzuaddiert. Um ein Element adressieren zu können, enthält jedes Element einen Vergleicher, mit dem überprüft wird, ob die Adresse des Elements mit einer von extern (von einer Ladelogik) oder intern gelieferten Adresse, die ein Element eines Bausteins des Clusters zur Umkonfiguration o. ä. ansprechen soll, übereinstimmt. In diesem Vergleicher wird die X- und Y-Adresse des Elements mit der zur Adressierung anliegenden X- und Y-Adresse verglichen. Dies geschieht in jeweils einem Komparator oder in einem Komperator für beide Adressen zugleich. Die Ausgänge der Komparatoren werden über ein UND-Gatter verknüpft und bilden das Enable Signal für das konfigurierbare Element. Das heißt, das Element wird nur angesprochen, wenn das externe Adressignal (von einer Ladelogik) mit der automatisch generierten Adresse des Elements übereinstimmt.

Es wäre selbstverständlich möglich die Logik zur Berechnung der Adresse des letzten Elements in X- und Y-Richtung wegzulassen. Dazu muß die von dem letzten Element in X- und Y-Richtung berechnete Adresse herausgeführt und zum nächsten Baustein weitergeleitet werden.

Weiterhin wäre denkbar, den Cluster nicht nur in 2 Dimensionen (X- und Y-Richtung) aufzubauen, sondern noch die dritte (Z-Richtung), oder mehrere Dimensionen hinzuzunehmen. Jeder Baustein benötigt dann ein weiteres Register für jede Dimension, in dem die Anzahl der konfigurierbaren Elemente des Bausteins in dieser Dimension gespeichert wird, sowie einen weiteren Addierer für jede Dimension zur Berechnung der Adresse des letzten Elements in X- und Y-Richtung innerhalb des Bausteins in dieser Dimension.

Damit entsteht ein N-dimensionales Array. Diese weiteren Dimensionen, außer der Z-Richtung, besitzen natürlich keine physikalische Repräsentation mehr, sondern werden nur durch eine Erweiterung des Adressraumes auf N-Dimensionen beschrieben. Dieselbe Erweiterung des Adressraumes auf N-Dimensionen ist auch für die konfigurierbaren Elemente innerhalb eines Bausteins denkbar.

3. Kurzbeschreibung der Diagramme

Fig. 1

a) Baustein mit automatischer Adressgenerierung.
b) Logik zur Adressgenerierung mit parallelem Eingangssignal und Ausgangssignal.
c) Logik zur Adressgenerierung mit seriellem Eingangssignal und Ausgangssignal.
d) Verschaltung zweier hintereinander folgender Logiken zur Adressgenerierung.
e) Signalverlauf der Daten- und Enable Signale bei der Verschaltung zweier Logiken zur Adressgenerierung.

Fig. 2 Baustein mit automatischer Adressgenerierung und mehreren konfigurierbaren Elementen.
Fig. 3

a) Logik zur Adressgenerierung und Adressierung eines Elements innerhalb eines Bausteins, bei paralleler Über-

tragung der Adresse.
b) Logik zur Adressgenerierung und Adressierung eines Elements innerhalb eines Bausteins bei serieller Übertragung.
c) Serieller Addierer

Fig. 4 Cluster aus mehreren Bausteinen und ihre Verschaltung miteinander.
Fig. 5

a) Baustein mit mehreren konfigurierbaren Elementen und einer alternativen Art der Adressierung.
b) Logik zur Adressierung eines Elements innerhalb eines Baustein bei alternativer Adressierung.

Fig. 6 Cluster mit nicht vollständigen Reihen oder Spalten.

4. Detailbeschreibung der Diagramme

Fig. 1a) zeigt einen Baustein **0109** eines Clusters. Über die Verbindung **0101** erhält er die X-Position des letzten Elements in X-Richtung vom vorherigen Baustein. Die Y-Position wird über die Verbindung **0102** in den Baustein übertragen. Im Register **0105** wird die Anzahl der Elemente in X-Richtung gespeichert, anschließend wird mit Hilfe eines Addierers **0106** zu dieser Anzahl die an Verbindung **0101** anliegende Adresse hinzuaddiert. Mit der Y-Adresse wird in analoger Weise verfahren. Das Register **0103** speichert die Anzahl der Elemente des Bausteins in Y-Richtung und im Addierer **0104** wird die über die Verbindung **0102** anliegende Adresse hinzuaddiert. Diese beiden neuen, automatisch generierten Werte bilden die X- und Y-Basisadresse des in X- und Y-Richtung nachfolgenden Bausteins. Sie stehen über die Verbindung **0107** für die X-Adresse, **0108** für die Y-Adresse dem nachfolgenden Baustein zur Verfügung. Die Verbindungen **0110** und **0111** dienen zur Übertragung der Y- und X-Adresse innerhalb des Bausteins, um für die Berechnung der Adressen der konfigurierbaren Elemente zur Verfügung zu stehen.

b) zeigt die Logik bei parallel übertragenem Eingangs- und Ausgangssignal, um die Adresse des letzten Elements in X-Richtung (Y-Richtung) des Bausteins zu berechnen. Über den Bus **0112** werden die Daten der Anzahl der konfigurierbare Elemente des Bausteins in X-Richtung (Y-Richtung) aus dem Register **0114** zum Addierer **0115** übertragen. Dort wird die Adresse des letzten Elements des vorhergehenden Baustein in X-Richtung (Y-Richtung) **0113** hinzuaddiert. Vom Addierer **0115** wird die berechnete Adresse über den Bus **0116** zum nächsten Baustein weitergegeben.

c) zeigt die Logik zur Speicherung und Berechnung der Adresse bei einem seriellen Eingangs- und Ausgangssignal. Die Daten der Adresse des vorhergehenden Bausteins in X-Richtung (Y-Richtung) werden seriell über eine Leitung **0118** übertragen. Über die Leitung **0119** wird das Taktsignal des Bausteins übertragen. Der serielle Datenstrom der Leitung **0118** gelangt zum Addierer **0124**, der die im Baustein gespeicherte Anzahl der Elemente sequentiell hinzuaddiert. Dazu wird die in Register **0121** gespeicherte Anzahl der Elemente beim Start in das Schieberegister **0120** geladen. Das Schieberegister **0120** wird wiederum durch den Takt **0119** gesteuert, so daß es die einzelnen Bits der Adresse synchron zu den Bits des Datenstroms auf Leitung **0118** zum Addierer **0124** schickt. Die Daten des Schieberegisters werden mit der positiven Taktflanke weitergeschoben. Im Addierer **0124** werden die einzelnen Bits nacheinander addiert und wieder sequentiell zum Ausgangsflipflop **0123** weitergeschickt. Über die Leitung **0125** werden diese Daten zum nächsten Baustein übertragen. Das Flipflop **0126** dient zur Speicherung des bei der Addition der beiden Bits anfallenden Übertrags, der bei der Addition der nächsten beiden Bits wieder berücksichtigt werden muß. Das Ausgangsflipflop **0123** und das Flipflop **0126** übernehmen Daten mit der negativen Taktflanke. Über die Verbindung **0117** wird ein Enable Signal zu den Flipflops **0122, 0123, 0126** und dem Schieberegister **0120** geleitet. Das Flipflop **0122** übernimmt das Enable Signal mit der negativen Taktflanke und gibt es über die Leitung **0127** zu den nachfolgenden Bausteinen weiter.

Die Datenübertragung verläuft nun folgendermaßen. Die Daten werden über die Leitung **0118** eingelesen. Gleichzeitig kommt über die Leitung **0117** ein Enable Signal, das dieselbe Länge wie die übertragenen Daten besitzt. Im Addierer **0124** wird nun die neue Adresse berechnet und über das Flipflop **0123** zum nächsten Baustein übertragen. Das Enable Signal wird über das Flipflop **0122** zum nächsten Baustein geleitet. Durch die Übernahme der Daten und des Enable Signals mit der negativen Flanke in die Flipflops, gelangen diese beiden Signale mit einem halben Takt Verzögerung zum nächsten Baustein.

d) zeigt anhand eines Timing-Diagramms wie zwei Logiken zur Adressgenerierung verschaltet werden. Diese Logiken sind in jeweils einem Baustein **0130** untergebracht. Beide Logiken haben denselben Aufbau aus einem Addierer **0134, 0140** mit Flipflop **0131, 0142** zur Speicherung des Carry Bits, einem Ausgangsflipflop **0135, 0141**, einem Flipflop zur Übernahme des Enable Signals **0136, 0143** sowie einem Register, in dem die Anzahl der Elemente des Bausteins gespeichert ist **0133, 0139** und einem Schieberegister **0132, 0138**. Der Unterschied zwischen den beiden Logiken besteht darin, daß die Flipflops **0141, 0142, 0143** und das Schieberegister **0138** der zweiten Logik mit der anderen Taktflanke gesteuert werden, als die Flipflops **0131, 0135, 0136** und das Schieberegister **0132** der ersten Logik. Dies kann wie in der Figur dargestellt durch die Invertierung von jedem Takteingang der Flipflops und des Schieberegisters oder durch die Invertierung des Taktsignals erfolgen. Damit ist sichergestellt, daß die Daten korrekt von den Flipflops und den Schieberegistern übernommen werden.

Bei der Verschaltung mehrerer Logiken (d. h. mehrere konfigurierbare Elemente hintereinander) wird die Taktflanke mit der die Daten in die Flipflops und die Schieberegister übernommen werden immer abgewechselt. Dadurch erfolgt eine korrekte Datenübernahme und das Datenpaket wird durch die Bausteine hindurchgeleitet.

e) zeigt den Signalverlauf des Daten- und Enable Signals für den Fall der Verschaltung mehrerer Logiken. Das Signal CLK **0144** stellt das Taktsignal des Bausteins dar. D1 **0145** und E1 **1046** stellen das Daten- und Enable Signal am Eingang der ersten Logik dar. D2 **0147** und E2 **0148** sind die Daten und Enable Signale am Ausgang der ersten Logik. Sie sind um einen halben Takt gegenüber den Signalen am Eingang verzögert, da sie von den Flipflops und dem Schieberegister erst mit der abfallenden Taktflanke übernommen werden (vgl. **Fig. 1d**). Gleichzeitig bilden sie die Eingangssignale

für die nachfolgende Logik zur Adressberechnung. Dort werden sie wieder um einen halben Takt verzögert und gelangen zum Ausgang, wo sie die Signale D3 **0149** und E3 **0150** bilden.

Fig. 2 stellt einen Baustein **0212** eines Clusters mit mehreren konfigurierbaren Elementen **0211** dar. Jedes dieser Elemente **0211** besitzt konfigurierbare Zellen und zur Konfiguration verwendete Elemente **0210**. Außerdem enthält jedes Element **0211** einen Vergleicher nach PACT02 **Fig. 3 0301** und eine Logik zur Berechnung der Elementadresse **0209**, wobei der Vergleicher für die Dekodierung der Adresse bei einem Zugriff auf ein Element zuständig ist. Damit wird die Adressierung der einzelnen Elemente eines Bausteins ermöglicht. Zusätzlich enthält der Baustein eine Logik zur automatischen Adressgenerierung **0207**, die wie in **Fig. 1** ausgeführt ist. Die Y-Adresse des letzten Elements des vorhergehenden Bausteins in Y-Richtung gelangt über den Bus **0201** in den Baustein **0212** und wird in der Logik zur automatischen Adressgenerierung **0207** weiterverarbeitet. Über die Verbindung **0205** wird diese Adresse zu den Elementen **0211** der ersten Reihe des Bausteins übertragen. Die Leitung **0203** dient zur Übertragung der Y-Adresse des letzten Elements dieses Bausteins in Y-Richtung zum nächsten Baustein in Y-Richtung. Die X-Adresse des letzten Elements des vorhergehenden Bausteins in X-Richtung gelangt über die Verbindung **0202** in den Baustein **0212** und wird in der Logik zur automatischen Adressgenerierung **0208** weiterverarbeitet. Sie wird über die Verbindung **0206** zu den Elementen **0211** der ersten Spalte des Bausteins **0212** übertragen. Die Leitung **0204** dient zur Übertragung der X-Adresse des letzten Elements dieses Bausteins zum nächsten Baustein.

Über den Bus **0213** gelangt die Adresse des Elements, das für eine Umkonfigurierung o. ä. angesprochen werden soll, zu den Zellen **0211**.

Bei der Adressierung wäre es auch möglich auf die Logik zur automatischen Adressgenerierung **0207, 0208** zu verzichten. Dazu muß die Y-Adresse des letzten Elements des Bausteins und die X-Adresse des letzten Elements des Bausteins herausgeführt werden, um die Adressdaten an den nächsten Baustein weiterzugeben. Diese Signale ersetzen dann die Signale **0203, 0204**.

Fig. 3a) zeigt den Aufbau des Vergleichers und der Adressgenerierung der Elemente **0211** aus **Fig. 2** bei paralleler Übertragung der Adressdaten. Zur der über die Verbindung **0302** anliegende X-Adresse des vorhergehenden Elements wird im Addierer **0305** eine Eins **0303** hinzuaddiert. Diese X-Adresse wird im Komparator **0309** mit der von außen kommenden X-Adresse **0307** verglichen. Im Addierer **0306** wird der über die Verbindung **0301** anliegende Y-Adresse des vorhergehenden Elements eine Eins **0304** hinzuaddiert. Anschließend vergleicht der Komparator **0310** den Wert mit der über die Verbindung **0308** anliegenden Y-Adresse des zu adressierenden Elements. Über ein UND-Gatter **0311** wird das Signal **0312** zur Aktivierung des Elements erzeugt. Die Verbindung **0314** dient zur Übertragung der berechneten X-Adresse zum nachfolgenden Element. Die Y-Adresse wird über die Verbindung **0313** übertragen.

b) zeigt den Aufbau des Vergleichers und der Adressgenerierung der Elemente 0211 aus **Fig. 2** bei serieller Übertragung der Adressdaten. Dazu wird zum seriellen Bitstrom der Adresse des vorhergehenden Elements in X-Richtung **0315** in der Logik zur Adressgenerierung **0317** eine Eins seriell hinzuaddiert und die Adresse in einem Schieberegister gespeichert. Die Logik bekommt das Taktsignal des Bausteins durch die Leitung **0327**. Über die Leitung **0321** wird die Adresse seriell zum nachfolgenden Element übertragen. Im Komparator **0319** wird die in **0317** gespeicherte Adresse mit der über die Leitung **0325** anliegenden X-Adresse des zu adressierenden Elements verglichen. Das Ganze geschieht analog für die Y-Adresse. Der von der vorhergehenden Zelle in Y-Richtung seriell übertragene Y-Adresse **0316** wird in der Logik zur Adressgenerierung **0318** eine Eins hinzuaddiert. Die Logik bekommt das Taktsignal des Bausteins durch die Leitung **0328**. Über die Verbindung **0322** wird die Adresse seriell zum nachfolgenden Element in Y-Richtung transportiert. Im Komparator **0320** wird die in **0318** gespeicherte Adresse mit der über die Verbindung **0326** anliegende Y-Adresse des zu adressierenden Elements verglichen. Die Ausgänge der beiden Komparatoren **0319, 0322** werden über ein UND-Gatter **0323** verknüpft und bilden das Enable Signal **0324** für das konfigurierbare Element.

c) zeigt die Logik zur Adressgenerierung **0317, 0318** aus **Fig. 3b**. Die Adresse gelangt über die Leitung **0329** zum Addierer **0332**. Das Flipflop **0334** dient zum Speichern des Übertrags, der bei der Addition entsteht. Es übernimmt den Übertrag immer mit der negativen Flanke des Taktsignals **0330**. Außerdem ist dieses Flipflop anfangs auf eins gesetzt, so daß bei der Addition des ersten Bit eine Eins addiert wird. Die neu berechnete Adresse wird zum nachfolgenden Schieberegister **0337** und zum Ausgangsflipflop **0336** geschickt. Dort werden sie mit der negativen Taktflanke übernommen. Das Schieberegister **0337** speichert die einzelnen Bits, so daß am Ende der Übertragung die Adresse des Elements im Schieberegister gespeichert ist. Diese wird dann über die Leitung **0339** zum Komparator weitergegeben. Über die Leitung **0333** werden die Daten der Adresse seriell zum nachfolgenden, Element geschickt. Die Leitung **0331** überträgt ein Enable Signal an die Flipflops und das Schieberegister. Dieses Enable Signal läuft synchron mit dem Datensignal, das heißt wenn Daten gesendet werden wird gleichzeitig ein Enable Signal übertragen. Das Enable Signal wird vom Flipflop **0335** mit der negativen Taktflanke übernommen und von der Leitung **0338** zum nächsten Element übertragen. Die Datenübertragung verläuft wie in **Fig. 1c** beschrieben. Auch hier ist es erforderlich, wenn mehrere Logiken hintereinander geschaltet werden, den Takt oder die Takteingänge der Flipflops und Schieberegister bei jeder nachfolgenden Lokik zu invertieren, so daß die Daten immer abwechselnd mit der ansteigenden und der abfallenden Flanke übernommen werden.

Fig. 4 zeigt die Verschaltung mehrerer Bausteine **0401** zu einem Cluster. Über die Verbindung **0402** werden die Y-Basisadressen der Bausteine am oberen Rand des Clusters initialisiert. Die X-Basisadressen der Bausteine am linken Rand des Clusters werden über die Verbindung **0403** initialisiert. Bei der parallelen Adressierung genügt es die Basisadressen an die Eingänge anzulegen. Für die Adressierung mit serieller Datenübertragung muß das Datenpaket mit der Basisadresse und ein Enable Signal seriell in den Baustein übetragen werden. Den im Inneren des Clusters liegenden Bausteinen, werden nach dem in **Fig. 1** beschriebenen Verfahren automatisch X- und Y-Basisadressen über die Verbindungen zueinander zugewiesen. Die Verbindungen **0404** und **0405** dienen zur Übertragung der Größe des Clusters an externe Elemente. Die Verbindung **0404** dient zur Übertragung der Größe in X-Richtung und über die Verbindung **0405** wird die Größe des Clusters in Y-Richtung übertragen.

Fig. 5a) zeigt den Aufbau eines Bausteins **0501**, der eine Vielzahl von konfigurierbaren Elementen **0514**, die konfigurierbare und zur Konfiguration verwendete Zellen enthalten, sowie eine Logik zur automatischen Adressgenerierung **0506, 0508** enthält. Die Adressierung der Bausteine **0501** und konfigurierbaren Elemente **0514** erfolgt hierbei auf eine al-

ternative Art. Die Adresse besitzt folgenden Aufbau:

X-Pos Bausteins	X-Pos Element	Y-Pos Bausteins	Y-Pos Element

Die Daten der X- und Y-Adresse werden durch die Verbindungen **0502, 0503** in den Baustein **0501** übertragen. Außerdem werden sie noch in die Vergleicher **0507, 0509** weitergeleitet. Innerhalb des Baubsteins **0501** werden sie von der Logik zur automatischen Adressgenerierung **0506, 0508** weiterverarbeitet. Bei der Weiterverarbeitung wird zur Adresse des Bausteins **0501** jeweils eine Eins in X- und Y-Richtung hinzugezählt und nicht wie bei der vorherstehenden Adressierungsart die Anzahl der Elemente in X- und Y-Richtung des Bausteins **0501**. Durch die Verbindungen **0504, 0505** werden sie zum nachfolgenden Baustein weitergeleitet. Wird nun über die Leitung **0513** eine Adresse geschickt, so wird der Adressteil, der die Bausteine adressiert, an die Vergleicher **0507, 0509** geleitet. Der Teil der Adresse, der die Elemente **0514** adressiert, wird an die Elemente **0514** geleitet. In den Vergleichern **0507, 0509** wird nun die automatisch generierte Bausteinadresse mit der Bausteinadresse, die über die Leitung **0513** anliegt, verglichen und im Falle einer Übereinstimmung ein Signal zum UND-Gatter **0510** geschickt, das ein Enable Signal an die konfigurierbaren Elemente **0514** schickt. Die Vergleicher **0511** der Elemente **0514** vergleichen ihre Adresse mit der über die Verbindung **0513** anliegenden Adresse und aktivieren sie bei einer Übereinstimmung und einem anliegenden Enable Signal der Vergleicher **0507, 0509** des Bausteins.

b) zeigt die Logik zur Adressierung eines Elements (vgl. Vergleicher **0511**), die für das alternative Adressierungsverfahren benötigt wird. Die Verbindungen **0515** und **0516** dienen zur Übertragung einer X- und Y-Adresse, mit deren Hilfe ein Element angesprochen werden soll. Diese beiden Werte werden in den Komparatoren **0519** und **0520** mit den in den Registern **0517** und **0518** gespeicherten X- und Y-Adressen des Elements verglichen. Selbstverständlich können die Adressen der Elemente **0514** auch auf die zuvor beschriebene Art (vgl. **Fig. 3**) erzeugt werden. Die Ausgänge der beiden Komparatoren **0519** und **0520** werden mit dem UND-Gatter **0521** verknüpft. Der Ausgang des UND-Gatters **0521** wird mit dem Enable Signal **0523** der Vergleicher der Bausteinadresse (vgl. **Fig. 5a 0510**) über ein UND-Gatter **0522** verknüpft und bildet das Enable Signal **0524** für die Elemente **0514** des Bausteins **0501**.

Fig. 6 zeigt einen Cluster aus Bausteinen **0603**, dessen Spalten und Reihen nicht vollständig sind. Dadurch ist eine etwas andere Verschaltung der Bausteine **0603** nötig, die in der Figur dargestellt wird. Die Verbindungen **0601** und **0602** dienen zur Initialisierung der Bausteine der ersten Reihe und Spalte, da sie keine Vorgänger in dieser Richtung besitzen. Über die Verbindungen **0604** und **0605** kann die Ausdehnung des Clusters in X- und Y-Richtung abgefragt werden.

5. Begriffsdefinition

Cluster Mehrdimensionale vernetzte Anordnung von Bausteinen oder -gruppen
DPGA Dynamisch konfigurierbare FPGA's. Stand der Technik
D-FlipFlop Speicherelement, welches ein Signal bei der steigenden Flanke eines Taktes speichert.
Elemente Sammelbegriff für alle Arten von in sich abgeschlossenen Einheiten, welche als Stück in einem elektronischen Baustein zum Einsatz kommen können. Elemente sind also:

- Konfigurierbare Zellen aller Art
- Cluster
- RAM-Blöcke
- Logik
- Rechenwerke
- Register
- Multiplexer
- I/O Pins eines Chips

Flag (Fahne). Statusbit in einem Register, das einen Zustand anzeigt.
FPGA Programmierbarer Logikbaustein. Stand der Technik.
Gatter Gruppe von Transistoren, die eine logische Grundfunktion durchführen. Grundfunktionen sind z. B. NAND, NOR, Transmission-Gates.
H-Pegel Logisch 1 Pegel, abhängig von der verwendeten Technologie
Kantenzelle Zelle am Rand eines Zellarrays, oftmals mit direktem Kontakt zu den Anschlüssen eines Bausteines.
konfigurierbares Element. Ein konfigurierbares Element stellt eine Einheit eines Logik-Bausteines dar, welche durch ein Konfigurationswort für eine spezielle Funktion eingestellt werden kann. Konfigurierbare Elemente sind somit, alle Arten von RAM-Zellen, Multiplexer, Arithmetische logische Einheiten, Register und alle Arten von interner und externer Vernetzungsbeschreibung etc.
konfigurierbare Zelle. Siehe Logikzellen
Konfigurieren Einstellen der Funktion und Vernetzung einer logischen Einheit, einer (FPGA)-Zelle oder einer PAE (vgl. umkonfigurieren).
Konfigurationsdaten Beliebige Menge von Konfigurationsworten.
Konfigurationsspeicher. Der Konfigurationsspeicher enthält ein oder mehrere Konfigurationsworte.
Konfigurationswort. Ein Konfigurationswort besteht aus einer beliebig langen Bit-Reihe. Diese Bit-Reihe stellt eine gültige Einstellung für das zu konfigurierende Element dar, so daß eine funktionsfähige Einheit entsteht.
Ladelogik Einheit zum Konfigurieren und Umkonfigurieren der PAE. Ausgestaltet durch einen speziell an seine Aufgabe angepaßten Mikrokontroller.
Latch Speicherelement, das ein Signal für gewöhnlich während des H-Pegels transparent weiterleitet und während des L-

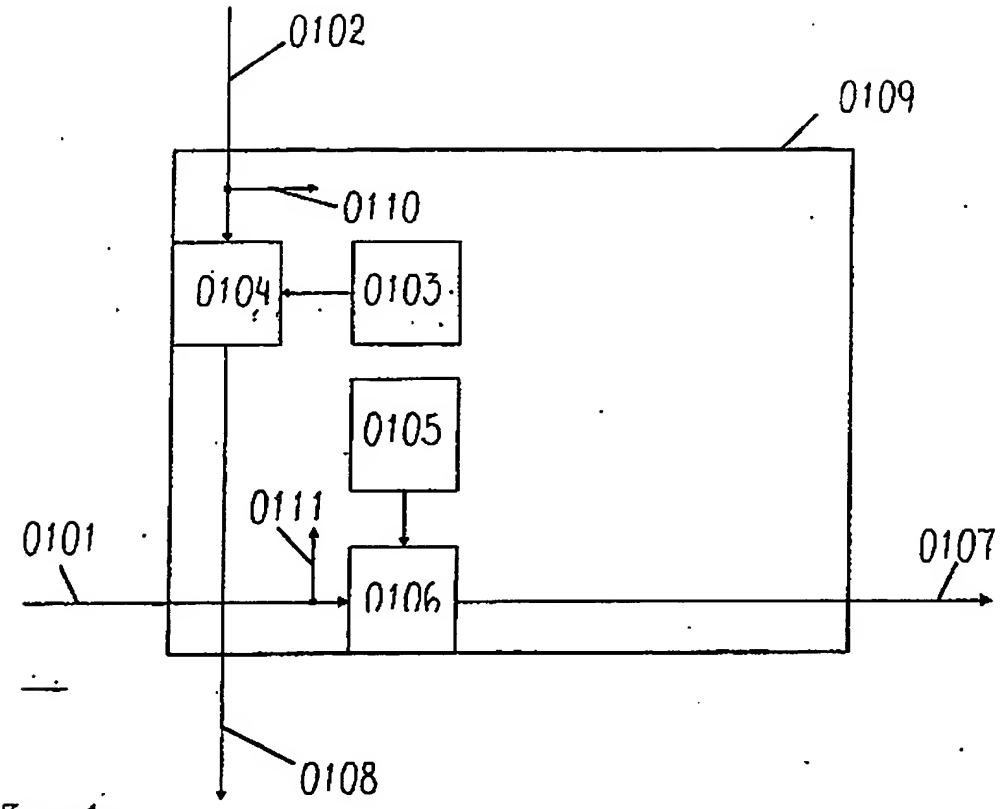
Pegels speichert.
In PAEs werden teilweise Latches gebraucht, bei denen die Funktion der Pegel genau umgekehrt ist. Hierbei wird vor den Takt eines üblichen Latch ein Inverter geschaltet.
Logikzellen: Bei DFPs, FPGAs, DPGAs verwendete konfigurierbare Zellen, die einfache logische oder arithmetische Aufgaben gemäß ihrer Konfiguration erfüllen.
L-Pegel Logisch 0 Pegel, abhängig von der verwendeten Technologie
Open-Kollektor Schaltungstechnik, bei der der Kollektor eines Transistors an einem, über einen Pullup auf den H-Pegel gezogenen, Bussignal liegt. Der Emitter der Transistors liegt auf Masse. Schaltet der Transistor, so wird das Bussignal auf den L-Pegel gezogen. Vorteil des Verfahrens ist, daß eine Mehrzahl solcher Transistoren den Bus ohne elektrische Kollision steuern können. Dabei sind die Signale ODER-verknüpft, es entsteht das sog. wired-OR.
RS-FlipFlop Reset-/Set-FlipFlop. Speicherelement, das durch 2 Signale umgeschaltet werden kann.
serielle Operationen Operationen, die durch serielles Abarbeiten eines Datenwortes oder eines Algorithmus durchgeführt werden. Serielle Multiplikation, serielle Division, Reihenentwicklung
Umkonfigurieren: Neues Konfigurieren von einer beliebigen Menge von PAEs, während eine beliebige Restmenge von PAEs ihre eigenen Funktionen fortsetzen (vgl. konfigurieren).
Zellen siehe Logikzellen.

Patentansprüche

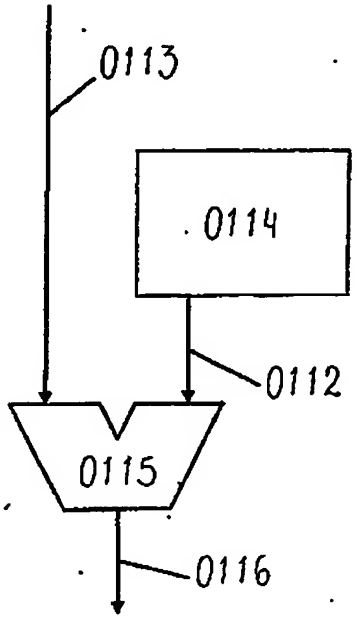
1. Verfahren zur dynamischen Adressgenerierung von konfigurierbarer Bausteinen, mit einer zwei oder mehrdimensionalen Zellanordnung (zum Beispiel FPGAs, DPGAs, DFPs o. ä.) und deren Zellen (konfigurierbaren Elemente), **dadurch gekennzeichnet**, daß
1. jeder Baustein einen Adresseingang für jede Dimension besitzt, durch den er seine Basisadresse, die die Adresse der ersten Zelle darstellt, erhält,
2. anhand der Basisadresse die baustein-internen Zelladressen (Adressen der konfigurierbaren Elemente) berechnet werden, über die eine Ladelogik die konfigurierbaren Elemente anspricht,
3. über Adressausgänge die Adresse der letzten Zelle plus eins, sowie je nach Ausgestaltung (siehe Unteransprüche) die Bausteinadresse plus eins, an die Nachfolgebausteine weitergegeben wird und deren Grundadresse darstellt,
2. Verfahren nach Anspruch 1, dadurch gekennzeichnet, daß die Adressein-/ausgänge als parallele Busse ausgestaltet sind.
3. Verfahren nach Anspruch 1, dadurch gekennzeichnet, daß die Adressein-/ausgänge als serielle Busse ausgestaltet sind.
4. Verfahren nach Anspruch 1–3, dadurch gekennzeichnet, daß die Eingangswerte in den Zellen des Zellarrays in jede Richtung weiterübertragen werden, wobei jede Zelle den Wert der Adresse um eins erhöht (vgl. Fig. 2).
5. Verfahren nach Anspruch 1–4, dadurch gekennzeichnet, daß je ein Addierer die Mächtigkeit des Zellarrays in der jeweiligen Richtung auf den Eingangswert plus eins aufaddiert und direkt an den Ausgang weitergibt (vgl. Fig. 1a).
6. Verfahren nach Anspruch 1–4, dadurch gekennzeichnet, daß die Eingangswerte in den Zellen des Zellarrays in jede Richtung weiterübertragen werden, wobei jede Zelle den Wert der Adresse um eins erhöht und der Ausgang der von der ersten Zelle am weitest entferntesten Zelle, also der Zelle mit der höchsten Adresse in alle Richtungen, direkt auf die Adressausgänge des Bausteines geschaltet wird (vgl. Beschreibung von Fig. 2).
7. Verfahren nach Anspruch 1, 3, 4–6, dadurch gekennzeichnet, daß parallele Addierer verwendet werden (Fig. 1c–e).
8. Verfahren nach Anspruch 1, 3, 4–6, dadurch gekennzeichnet, daß serielle Addierer verwendet werden (Fig. 1c–e).
9. Verfahren nach Anspruch 1–8 dadurch gekennzeichnet, daß die Adressen in jeder Zelle über Vergleicher mit der von der Ladeeinheit generierten Adresse verglichen wird, um einen Zugriff festzustellen (vgl. Fig. 3a, b, 5b).
10. Verfahren nach Anspruch 1–9 dadurch gekennzeichnet, daß die Adressen linear vergeben werden, wodurch ein linearer Adressraum aus Zellen über einer Gruppe von kaskadierten Bausteinen entsteht (vgl. Fig. 3).
11. Verfahren nach Anspruch 1–9 dadurch gekennzeichnet, daß die Adressen aus einem Offset für die Bausteinen und einer linearen Adresse für die Zellen eines Bausteines bestehen, wodurch kein linearer Adressraum aus Zellen über einer Gruppe von kaskadierten Bausteinen entsteht (vgl. Fig. 5).

Hierzu 9 Seite(n) Zeichnungen

Nummer: DE 197 04 044 A1
Int. Cl.[6]: G 06 F 9/44
Offenlegungstag: 13. August 1998

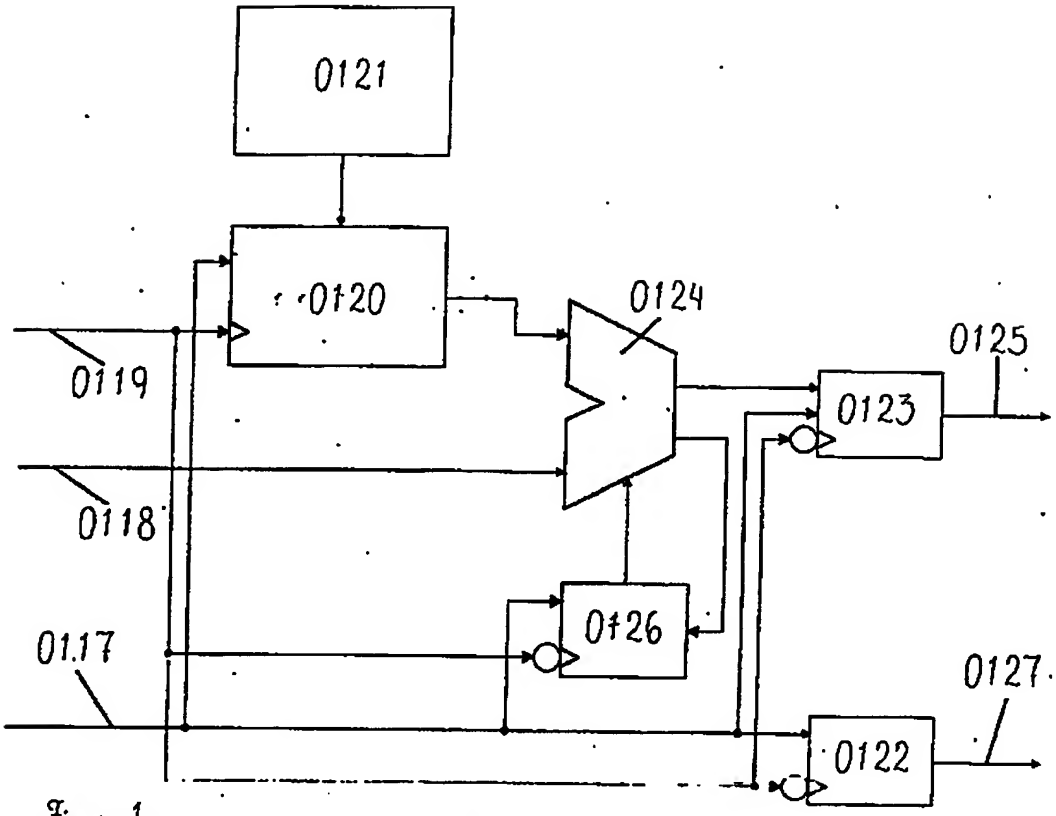


Figur 1a

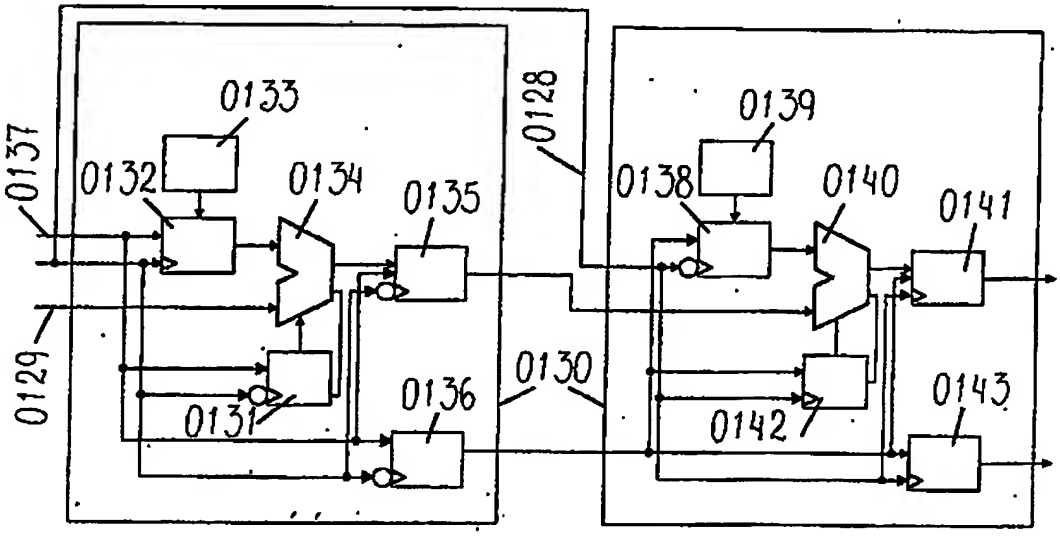


Figur 1b

Nummer: DE 197 04 044 A1
Int. Cl.[6]: G 06 F 9/44
Offenlegungstag: 13. August 1998

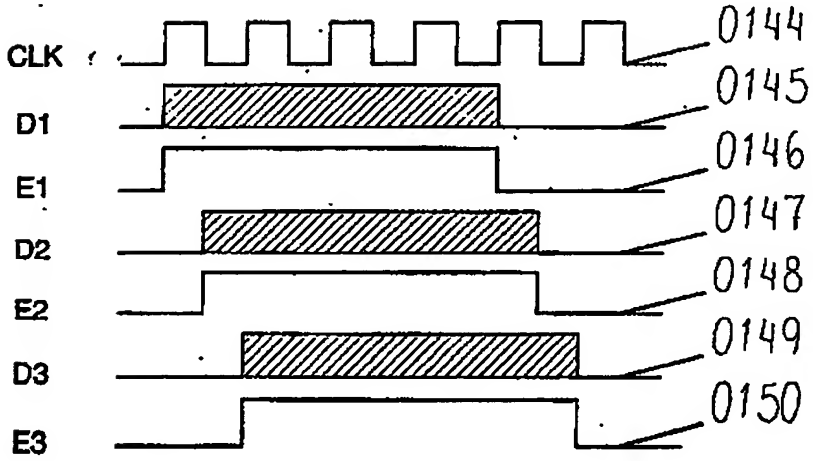


Figur 1c



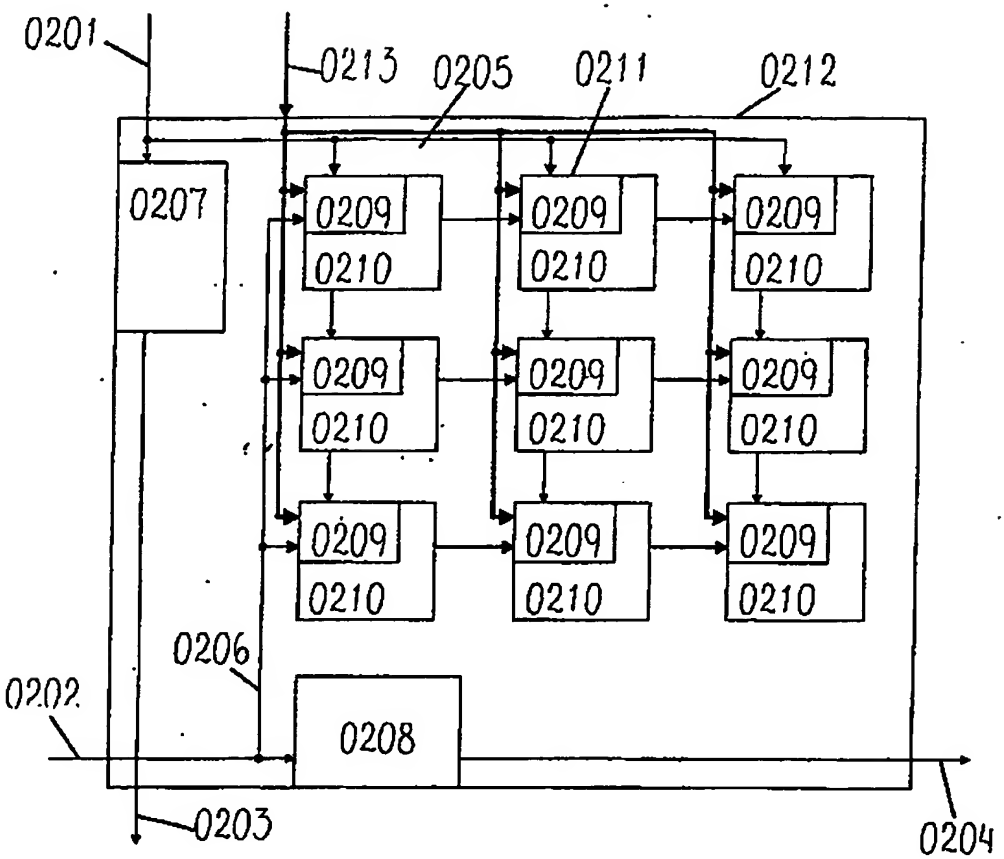
Figur 1d

Nummer: DE 197 04 044 A1
Int. Cl.[6]: G 06 F 9/44
Offenlegungstag: 13. August 1998



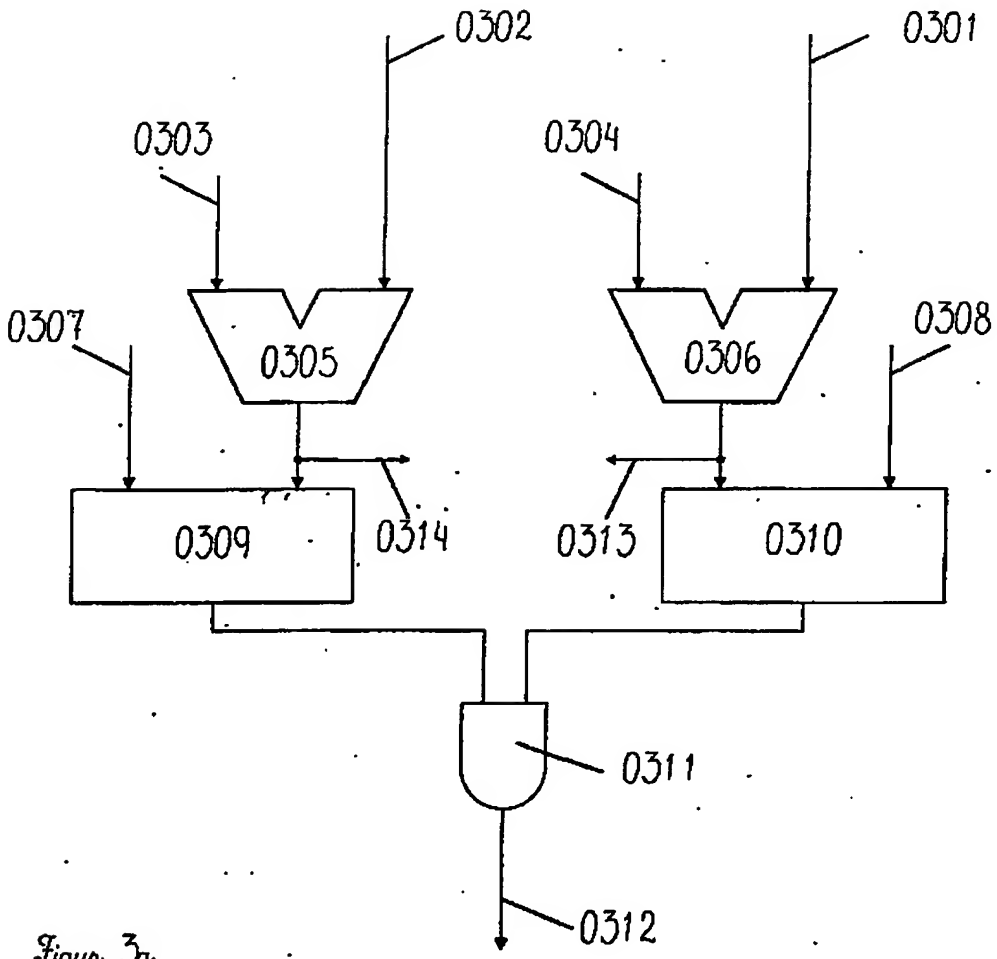
Figur 1e

Nummer: DE 197 04 044 A1
Int. Cl.[6]: G 06 F 9/44
Offenlegungstag: 13. August 1998

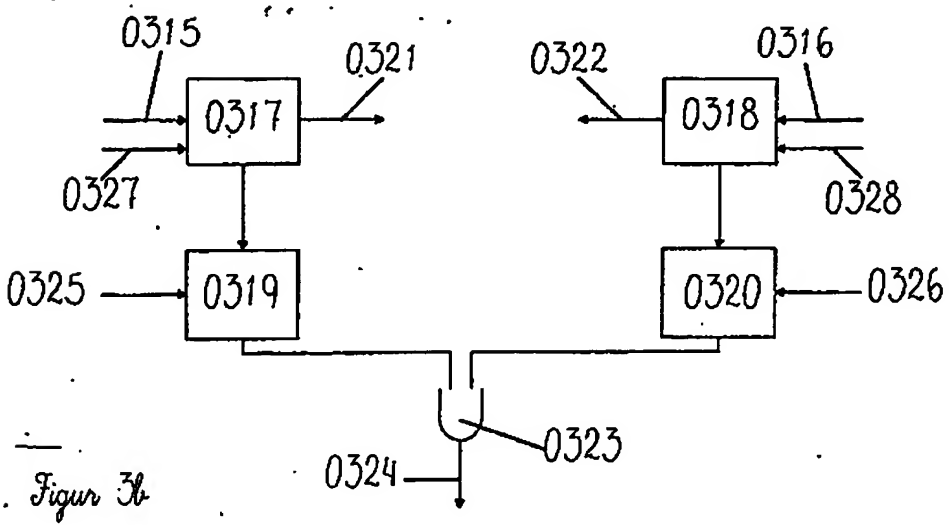


Figur 2

Nummer: DE 197 04 044 A1
Int. Cl.[6]: G 06 F 9/44
Offenlegungstag: 13. August 1998

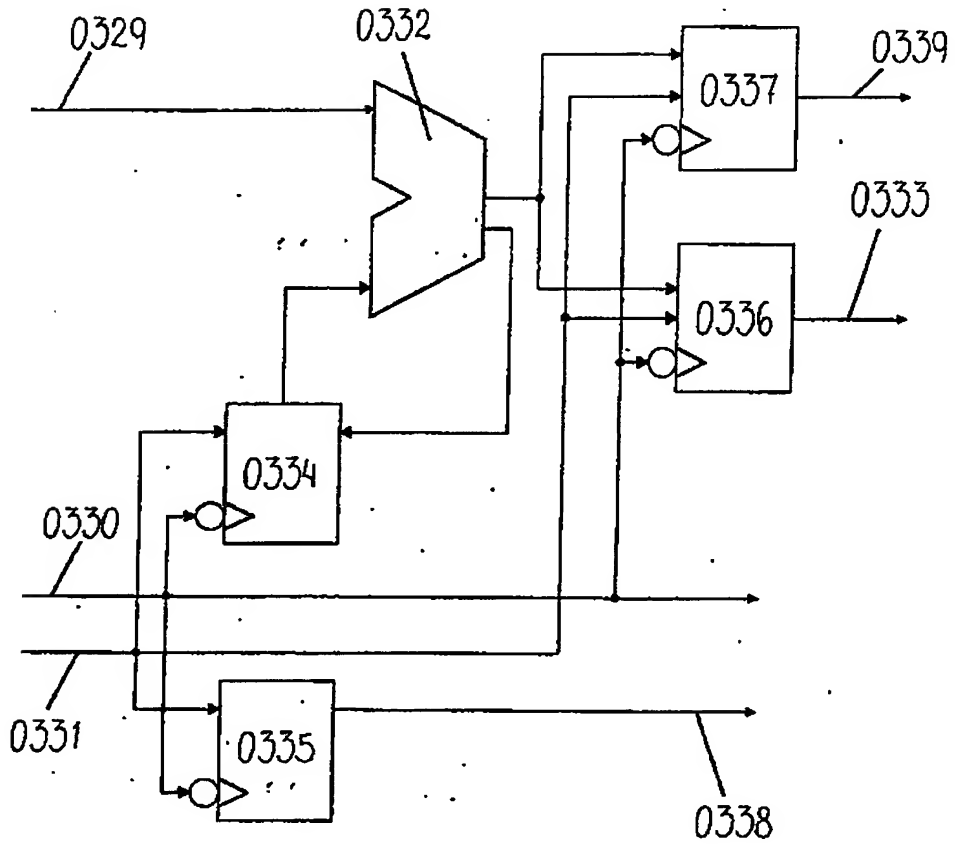


Figur 3a



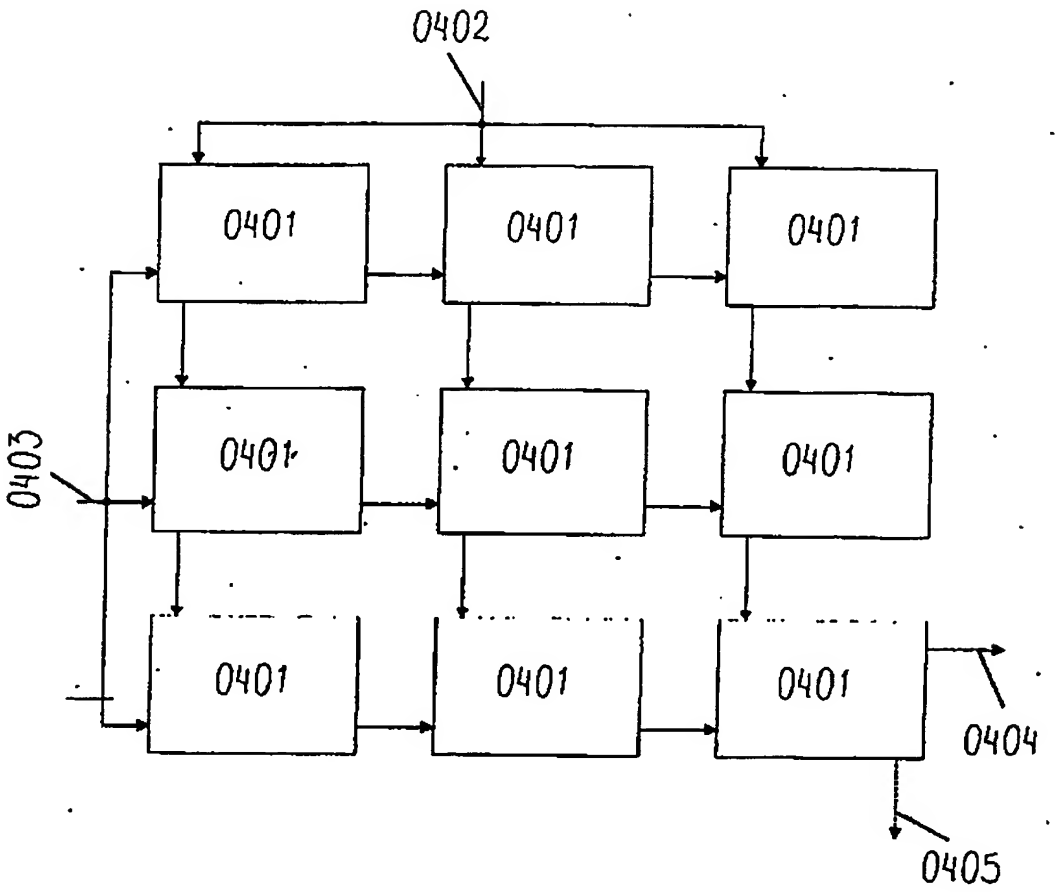
Figur 3b

Nummer: DE 197 04 044 A1
Int. Cl.[6]: G 06 F 9/44
Offenlegungstag: 13. August 1998



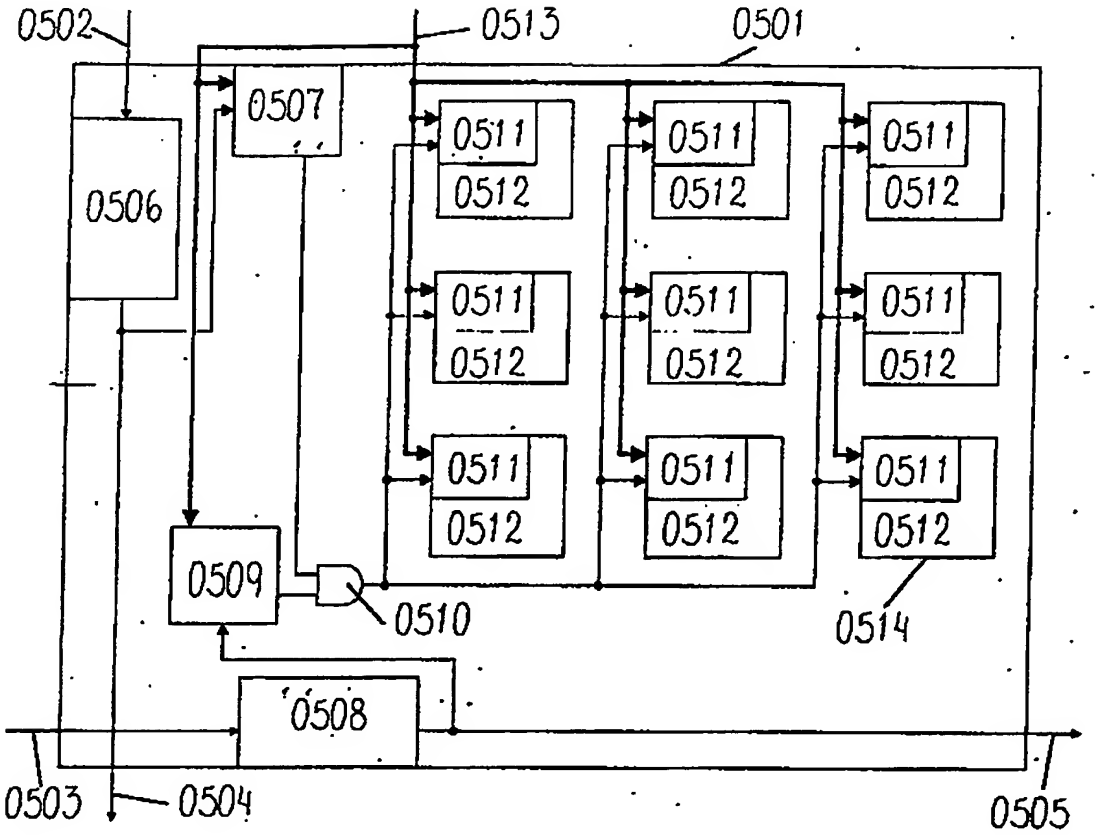
Figur 3c

Nummer: DE 197 04 044 A1
Int. Cl.[6]: G 06 F 9/44
Offenlegungstag: 13. August 1998

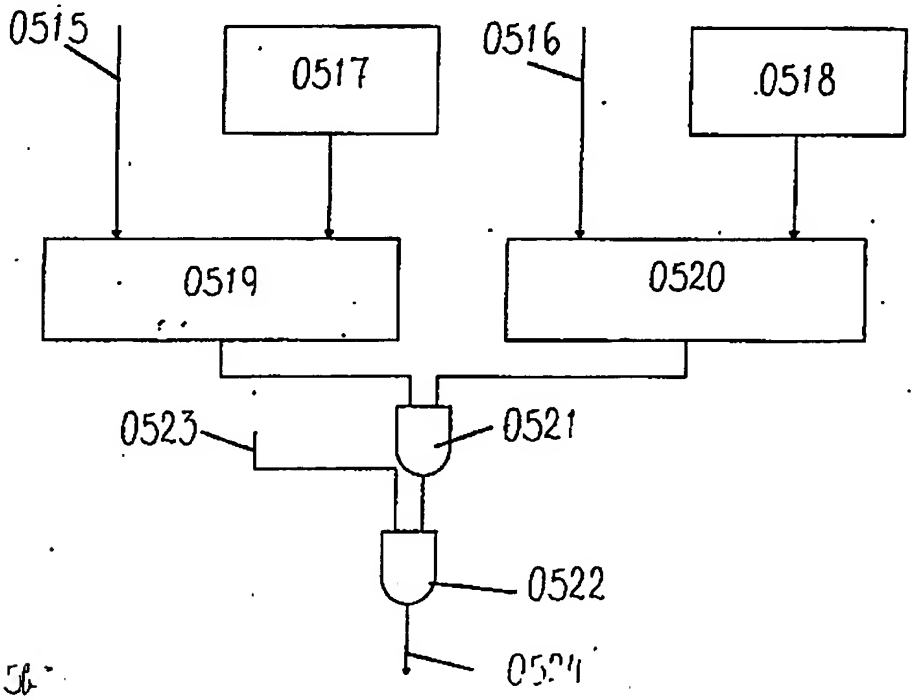


Figur 4

Nummer: DE 197 04 044 A1
Int. Cl.[6]: G 06 F 9/44
Offenlegungstag: 13. August 1998

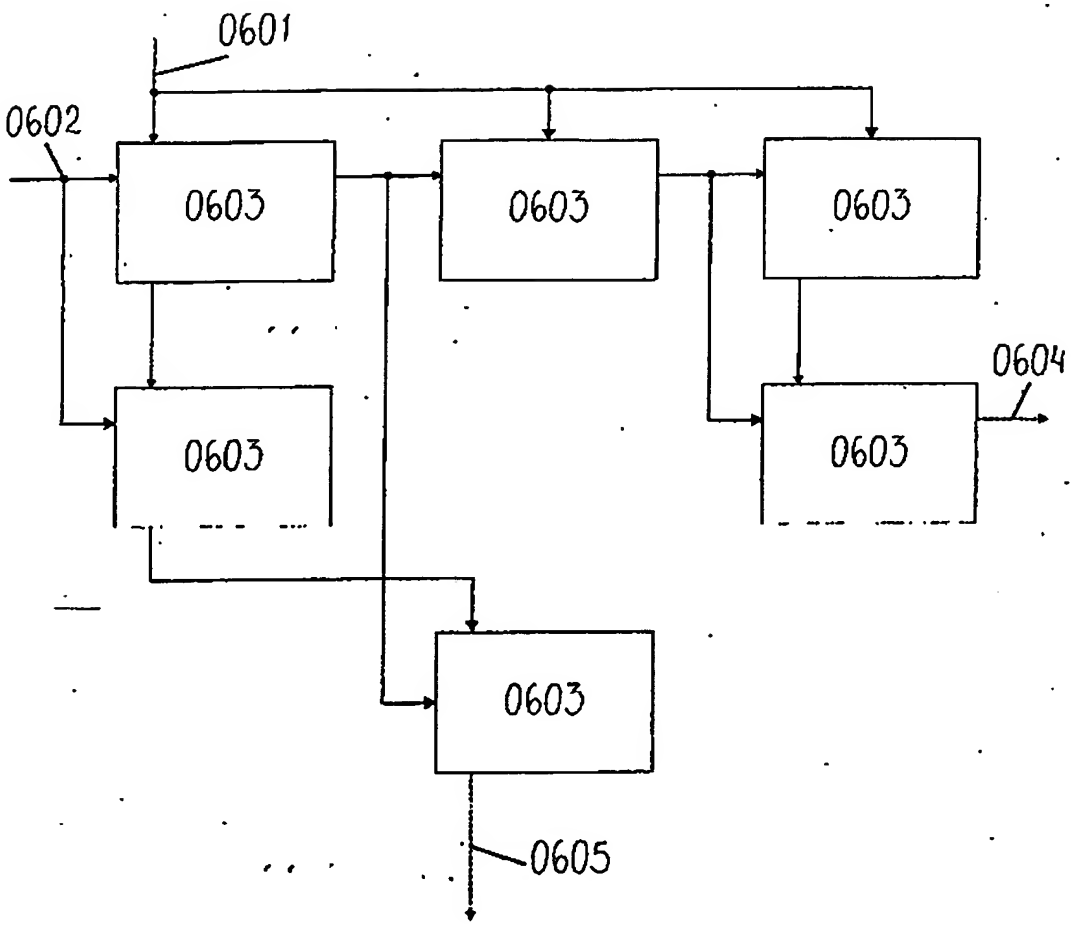


Figur 5a



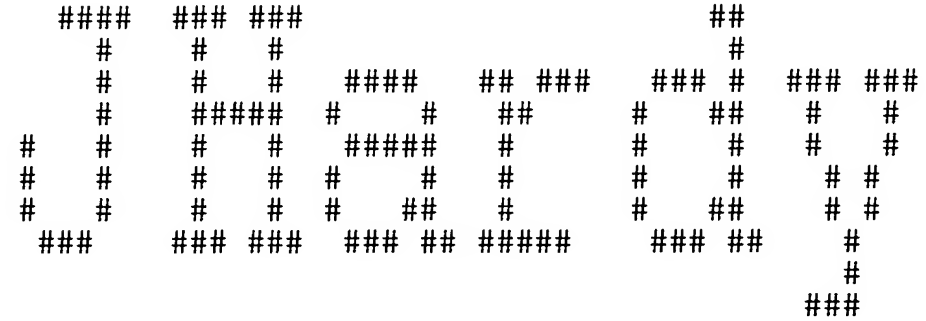
Figur 5b

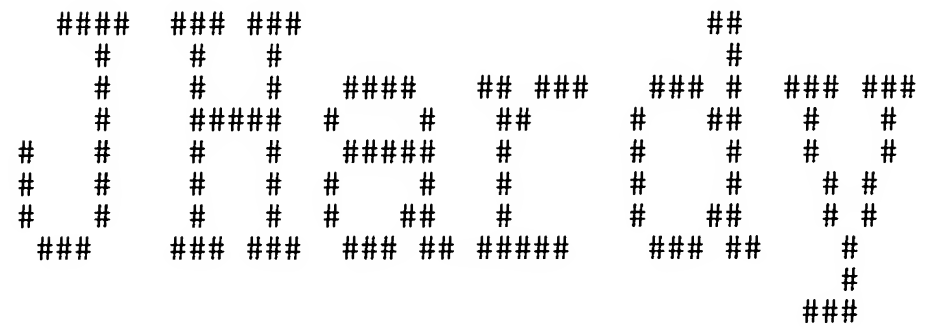
Nummer: DE 197 04 044 A1
Int. Cl.[6]: G 06 F 9/44
Offenlegungstag: 13. August 1998



Figur 6

****
****





****
****

Print Job Information:

Date: 11/14/2006

Time: 12:37:59 PM

Job Number: 778

Run-time reconfiguration method for programmable units

Patent number:	DE19654593
Publication date:	1998-07-02
Inventor:	MUENCH ROBERT (DE); VORBACH MARTIN (DE)
Applicant:	PACT INF TECH GMBH (DE)
Classification:	
- international:	G11C16/02; G06F9/445
- european:	G06F9/24, G06F15/78R
Application number:	DE19961054593 19961220
Priority number(s):	DE19961054593 19961220

Also published as:

- WO9831102 (A1)
- EP0947049 (A1)
- US6021490 (A1)
- EP0947049 (B1)

Abstract of **DE19654593**

The invention relates to a method for reconfiguration during the running time of FPGA, in which there is a loading logic or several loading logics which react to signals of any kind and recognize and can process special loading logic commands within a configuration programme consisting of data and commands, and, on the basis of the source of an event, can compute an entry in a branch table. For this, there are one or more branch tables for locating the address of the configuration data to be loaded after computing. One or more configuration memory areas exist, in which one or more configuration programmes are loaded; and there are one or more FIFO memory areas into which configuration data is copied which could not be sent to the element or elements to be configured. When an event occurs, an address is computed in a branch table, based on the source of the event. FIFO memory area is provided and run through before each reloading, and, if the cell can not be reloaded, the configuration data is copied into it nearer the beginning; if the cell can be reloaded, the configuration data is transferred to the cell. The computed branch table entry is read-out, and the configuration data which is stored at the read-out address is loaded into the cell, or, if the cell cannot be reprogrammed, it is copied into the FIFO memory area.